Chapter 1 : Full text of "Java, Java, Java Object-Oriented Problem Solving"

This is a review of the 3RD EDITION, published December The book starts from scratch. You do not have to know anything about Java. What Morelli and Walde offer is an education that emphasises an object oriented programming mentality, from the very start.

Designing a Riddle Program The first step in the program-development process is making sure you understand the problem Fig. Thus, we begin by developing a detailed specification, which should address three basic questions: What exactly is the problem to be solved? How will the program be used? How should the program behave? In the real world, the problem specification is often arrived at through an extensive discussion between the customer and the developer. In an introductory programming course, the specification is usually assigned by the instructor. Problem Specification Design a class that will represent a riddle with a given question and answer. Problem Decomposition Most problems are too big and too complex to be tackled all at once. So, the next step in the design process is to divide the problem into parts that make the solution more manageable. Divide and conquer Note that there is some ambiguity here about how far we should go in decomposing a given program. This ambiguity is part of the design process. One useful design guideline for trying to decide what objects are needed is the following: For example, the key noun in our current problem is riddle, so our solution will involve an object that serves as a model for a riddle. The main task of this Java object will be simply to represent a riddle. Two other nouns in the specification are question and answer. Fortunately, Java has built-in String objects that represent strings of characters such as words or sentences. A good understanding of object-oriented design can come only after much design experience, but this is a good place to start. Object Design Once we have divided a problem into a set of cooperating objects, designing a Java program is primarily a matter of designing and creating the objects themselves. In our example, this means we must now design the features of our riddle object. For each object, we must answer the following basic design questions: What role will the object perform in the program? What data or information will it need? What actions will it take? What interface will it present to other objects? What information will it hide from other objects? For our riddle object, the answers to these questions are shown in Figure 1. A class defines the collection of objects that belong to it. This is the same as for real-world objects. Thus, Seabiscuit is a horsethat is, Seabiscuit is an object of type horse. Similarly, an individual riddle, such as the newspaper riddle, is a riddle. That is, it is an object of type Riddle. Design specification for the Riddle class. To store and retrieve a question and answer Attributes Information question: The role of the Riddle object is to model an ordinary riddle. Because a riddle is defined in terms of its question and answer, our Riddle object will need some way to store these two pieces of information. As we learned in Chapter 0, an instance variable is a named memory location that belongs to an object. In general, instance variables are used to store the information that an object needs to perform its role. Deciding on these variables provides the answer to the question "What information does the object need? A useful design guideline for actions of objects is the following: Looking for Verbs Choosing the behavior of an object is often a matter of looking for verbs in the problem specification. For this problem, the key verbs are set and retrieve. As specified in Figure 1. What actions will the object take? Each of the actions we have identified will be encapsulated in a Java method. As you will recall from Chapter 0, a method is a named section of code that can be invoked, or called upon, to perform a particular action. In the object-oriented approach, calling a method method invocation is the means by which interaction occurs among objects. Calling a method is like sending a message between objects. This is like sending the message "Give me your answer. In designing an object, we must decide which methods should be made available to other objects. This determines what interface the object should present and what information it should hide from other objects. Except for its interface, all other information maintained by each riddle should be kept "hidden" from other objects. For example, it is not necessary for other objects to know where a riddle object stores its question and answer. The fact that they are stored in variables named question and answer rather than variables named ques and ans is irrelevant to other objects. What interface will it present, and what information will it hide? Information Hiding An object should hide most of the details of its implementation. Taken together, these various design decisions lead to the specification shown in Figure 1. As our discussion has illustrated, we arrived at the decisions by asking and answering the right questions. In most classes the attributes variables are private. This is represented by a minus sign -. The figure shows that the Riddle class has two hidden or private variables for storing data and three visible or public methods that represent the operations it can perform. A UML class diagram representing the Riddle class. Data, Methods, and Algorithms Among the details that must be worked out in designing a riddle object is deciding what type of data, methods, and algorithms we need. There are two basic questions involved: What type of data will be used to represent the information needed by the riddle? How will each method carry out its task? Like other programming languages, Java supports a wide range of different types of data, some simple and some complex. What type of data will be used? In designing a method, you have to decide what the method will do. In order to carry out its task, a method will need certain information, which it may store in variables. In addition, it will have to carry out a sequence of individual actions to perform the task. This is called its algorithm which is a step-by-step description of the solution to a problem. And, finally, you must decide what result the method will produce. Thus, as in designing objects, it is important to ask the right questions: What specific task will the method perform? What information will it need to perform its task? What algorithm will the method use? What result will the method produce? Methods can be thought of as using an algorithm to complete a required action. The algorithm required for the Riddle constructor is very simple but also typical of constructors for many classes. It takes two strings and assigns the first to the question instance variable and then assigns the second to the answer instance variable. The algorithms for the other two methods for the Riddle class are even simpler. They are referred to as get methods that merely return, or produce, the value that is currently stored in an instance variable. Not all methods are so simple to design, and not all algorithms are so simple. Even when programming a simple arithmetic problem, the steps involved in the algorithm will not always be as obvious as they are when doing the calculation by hand. For example, suppose the problem were to calculate the sum of a list of numbers. If we were telling a classmate how to do this problem, we might just say, "Add up all the numbers and report their total. Set the initial value of the sum to 0. If there are no more numbers to total, go to step 5. Add the next number to the sum. Go to step 2. Algorithm design Each step in this algorithm is simple and easy to follow. It would be relatively easy to translate it into Java. Because English is somewhat imprecise as an algorithmic language, programmers frequently write algorithms in the programming language itself or in pseudocode, a hybrid language that combines English and programming language structures without being too fussy about programming-language syntax. For example, the preceding algorithm might be expressed in pseudocode as follows: But many programming problems are quite complex, and careful design is required to minimize the number of errors in the program. In such situations, pseudocode could be useful. Another important part of designing an algorithm is to trace itthat is, to step through it line by lineon some sample data. For example, we might test the list-summing algorithm by tracing it on the list of numbers shown here.

Chapter 2 : Exercises | Java, Java, Java, Object-Oriented Problem Solving (3rd Edition)

*Java, Java, Java, Object-Oriented Problem Solving (3rd Edition) [Ralph Morelli, Ralph Walde] on calendrierdelascience.com *FREE* shipping on qualifying offers. Functional and flexible, this guide takes an objects-first approach to Java programming and problem using games and puzzles.*

Method definition and method invocation. Local scope and class scope. Primitive type and reference type. Access method and constructor method. If b1 is true, then print "one"; otherwise, print "two. The method should print "Hello" if its parameter is true; otherwise, it should print "Goodbye. The method should return "Hello" if its parameter is true; otherwise it should return "Goodbye. This one returns a String. That one was a void method. The method should return a String that consists of the word "Hello" concatenated with the value of its parameter. For example, if you call this method with the expression hello "dolly" , it should return "hello dolly. For example, intro "first" should print, "On the first day of Christmas my true love gave to me. Then write a main method that calls the other methods to print the whole song. For example, if you called permute "a", "b", "c" , it would produce the following output: That is, create a limerick template that will take any five words or phrases and produce a limerick. For each of the following exercises, write a complete Java application program: The name can be any string, but the rating should be one of the following values: Define a CopyMonitor class that solves the following problem. A company needs a monitor program to keep track of when a particular copy machine needs service. The device has two important boolean variables: The servicing rule that the company uses is that service is needed when either , pages have been printed or the toner is too low. Pretend that the machine has other methods that keep track of toner level and page count. Write another method with the signature hadAnX String s , which sings the "had a duck" part of the verse, and a method withA String sound to sing the "with a quack quack here" part of the verse. Test your class by writing a main method. And suppose you have a subclass of A named B with methods named b , c , and d. Draw a UML diagram showing the relationship between these two classes. Explain the inheritance relationships between them and identify the methods that would be considered polymorphic. Define a subclass of C named B that overrides method m1 so that it returns the difference between m and n instead of their sum.

Chapter 3 : Morelli, Java, Java, Java Object-Oriented Problem Solving, 2nd Edition | Pearson

With a focus on Java's strengths and object-oriented problem solving, this revision of a popular book takes an "objects early" approach to teaching Java, with the assumption that teaching beginners the "big picture" early gives them more time to master the principles of object-oriented programming.

Chapter 4 : Morelli & Walde, Java, Java, Java, Object-Oriented Problem Solving, 3rd Edition | Pearson

This text takes an objects-first approach to programming and problem solving using Java. A flexible design allows instructors to choose between Command-Line Interface, Graphical User Interface (GUI), or file input and output.

Chapter 5 : Java, Java, Java: Object-Oriented Problem Solving, Third Edition

With a focus on Java's strengths and object-oriented problem solving, this revision of a popular book takes an 'objects early' approach to teaching Java, with the assumption that teaching beginners the 'big picture' early gives them more time to master the principles of object-oriented programming. pp. Englisch.

Chapter 6 : Java, Java, Java, Object-Oriented Problem Solving (3rd Edition) | eBay

DOWNLOAD PDF JAVA, JAVA, JAVA OBJECT-ORIENTED PROBLEM SOLVING (2ND EDITION)

This edition retains the "objects first" approach to programming and problem solving that was characteristic of the first two editions. Throughout the text we emphasize careful coverage of Java language features, introductory programming concepts, and object-oriented design principles.

Chapter 7 : Java, Java, Java: Object-Oriented Problem Solving 2/e

This second edition of Java, Java, Java offers a robust, accessible, and flexible problem-solving perspective. The use of Unified Modeling Language (UML) diagrams throughout the text, strongly emphasizes object-oriented design.

Chapter 8 : Java, Java, Java: Object-Oriented Problem Solving, 3rd Edition â€“ ScanLibs

For each of the following exercises, write a complete Java application program: Exercise Define a class named Donor that has two instance variables, the donor's name and rating, both of which are String s.

Chapter 9 : calendrierdelascience.com: Customer reviews: Java, Java, Java Object-Oriented Problem Solv

If you are looking for a book Java, Java, Java, Object-Oriented Problem Solving (3rd Edition) by Ralph Morelli, Ralph Walde in pdf format, in that case you come on to loyal site.